U.S.P.S. Express Mail Label No.: EL 848969285 US

Date of Deposit: August 22, 2003

ATTORNEY DOCKET NO. 14312US02

MOBILE HANDSET UPDATE PACKAGE GENERATOR THAT EMPLOYS NODES TECHNIQUE

CROSS-REFERENCE TO RELATED APPLICATIONS/INCORPORATION BY REFERENCE

[01] This patent application makes reference to, claims priority to and claims benefit from United States Provisional Patent Application Serial No. 60/405,253, entitled "Firmware Update Network And Process Employing Preprocessing Techniques," filed on August 22, 2002, United States Provisional Patent Application Serial No. 60/415,620, entitled "System for Generating Efficient And Compact Update Packages," filed on October 2, 2002, United States Provisional Patent Application Serial No. 60/441,867, entitled "Mobile Handset Update Package Generator That Employs Nodes Technique," filed on January 22, 2003, and United States Provisional Patent Application Serial No. 60/447,977, entitled "Update Package Generator Employing Partial Predictive Mapping Techniques For Generating Update Packages For Mobile Handsets," filed on February 18, 2003.

[02] The complete subject matter of each of the above-referenced United States Patent Applications is hereby incorporated herein by reference, in its entirety. In addition, this application makes reference to United States Provisional Patent Application Serial No. 60/249,606, entitled "System and Method for Updating and Distributing Information," filed November 17, 2000, and International Patent Application Publication No. WO 02/41147 A1, entitled "Systems And Methods For Updating And Distributing Information," publication date March 23, 2002, the complete subject matter of each of which is hereby incorporated herein by reference, in its entirety.

[03] This application is also related to the following co-pending applications, the complete subject matter of each of which is hereby incorporated herein by reference in its entirety:

Ser. No.	Docket No.	Title	Filed	Inventors
	14121US02	Firmware Update Network and Process Employing Preprocessing Techniques	August 21, 2003	Chen Gustafson
	14122US02	System for Generating Efficient and Compact Update Packages	August 21, 2003	Chen Gustafson Barber
	14911US02	Update Package Generator Employing Partial Predictive Mapping for Generating Update Packages for Mobile Handsets	August 21, 2003	Lilley

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[04] [Not Applicable]

[MICROFICHE/COPYRIGHT REFERENCE]

[05] [Not Applicable]

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[06] The present invention relates generally to update of firmware/software components in mobile handsets using an update agent, and, more specifically, to the generation of compact update packages employing optimization techniques.

BACKGROUND OF THE ART

[07] Electronic devices, such as mobile phones and personal digital assistants (PDAs), often contain firmware and application software that are either provided by the manufacturer of the electronic devices, by telecommunication carriers, or by third parties. These firmware

and application software often contain software bugs. New versions of the firmware/ software are periodically released to fix the bugs or to introduce new features, or both.

[08] When informing a mobile handset of the need to update its firmware/software, problems might arise. Problems may also arise when attempting to control the size of an update package containing the difference information. Update package generation is prone to an "avalanche" effect, wherein a minor change to code may result in relocation of a large amount of other code or components. The relocation of the code may result in difference information that is very large, thereby prolonging the download process and making it more costly. Other problems may arise, such as being able to clearly and efficiently specifying the steps needed to update firmware/software in the handset to a new version.

[09] Further limitations and disadvantages of conventional and traditional approaches will become apparent to one of ordinary skill in the art through comparison of such systems with the present invention as set forth in the remainder of the present application with reference to the drawings.

BRIEF SUMMARY OF THE INVENTION

[10] Aspects of the present invention may be seen in a mobile services network comprising a mobile electronic device having a non-volatile memory, a random access memory and security services; the network further comprising a management server; an update package repository; and a generator with nodes preprocessor for generating update packages for updating an old version of a firmware to a new version of the firmware. The update packages generated by the generator with nodes preprocessor may incorporate nodes and filter information.

[11] The process of generating an update package with the generator with nodes preprocessor comprises converting the symbols in the new and old images of the firmware into distance information; determining a list of nodes in the old and new images of the firmware; generating filter information; generating the update package; and outputting the update package.

[12] The nodes for the node list are determined by determining the addresses of the symbols in the old image and in the new image; comparing the differences in the addresses and predicting differences in addresses of subsequent symbols; determining the symbols for which offsets cannot be predicted; and using the symbols that could not be predicted as additional node symbols.

[13] These and other features and advantages of the present invention may be appreciated from a review of the following detailed description of the present invention, along with the accompanying figures in which like reference numerals refer to like parts throughout.

BRIEF DESCRIPTION OF THE DRAWINGS

[14] Fig. 1A illustrates a block diagram of an exemplary mobile services network, in accordance with an embodiment of the present invention.

[15] Fig. 1B illustrates a block diagram of exemplary changes to a binary memory image between two versions of firmware/software of a mobile handset, in accordance with an embodiment of the present invention.

[16] Fig. 2 illustrates a flow diagram of the operation of an exemplary generator with nodes preprocessor generating an update package using two versions of a firmware/software, in accordance with an embodiment of the present invention.

[17] Fig. 3 illustrates a flow diagram of the operation of another exemplary generator with nodes preprocessor generating an update package using two versions of a firmware/software, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[18] Fig. 1A illustrates a block diagram of an exemplary mobile services network 105, in accordance with an embodiment of the present invention. In the exemplary embodiment of Fig. 1, the mobile services network 105 comprises a mobile handset 107, a management server 109, an update package repository 133, and a generator with nodes preprocessor 145. The mobile handset 107 may have access to services that may include a firmware/software update service. The mobile handset 107 may retrieve update packages with associated nodes and filter information from the management server 109. The update packages may be generated by the generator with nodes preprocessor 145 and populated into the update package repository 133, which is linked to the management server 109. The update package, generated by the generator with nodes preprocessor 145, may incorporate nodes information and filter information within the same package. In another embodiment of the present invention, the mobile handset 107 may be a personal digital assistant (PDA) or a similar device.

[19] In one embodiment, the generator with nodes preprocessor 145 may be coupled to the update package repository 133. The management server 109 may access the update package repository 133 to retrieve the update packages and associated information, such as the metadata, nodes information, filter information, etc.

[20] In another embodiment, the generator with nodes preprocessor 145 may be located at a remote location. In such embodiment, the output of the generator with nodes preprocessor 145, namely update packages and associated information, may be communicated to the update package repository 133 through a storage medium, such as a CD-ROM disk, and loaded into the update package repository 133.

The generator with nodes preprocessor 145 may generate an update package and associated information, by comparing two different versions of the firmware/software of the mobile handset 107.

[21] The mobile handset 107 may comprise a non-volatile memory 111, a random access memory (RAM) 125, and security services 123. The non-volatile memory 111 of the mobile handset 107 may comprise an update agent 113, a firmware and real-time operating system (RTOS) 115, an operating system (OS) layer 117, a download agent or browser 119, an end-user-related data and content 121, and boot initialization 127.

[22] In one embodiment, the mobile handset 107 downloads an update package from the update package repository 133, and then reboots. Upon rebooting the mobile handset 107 starts up, executes the boot initialization 127, and determines whether the update agent 113 needs to execute the update process. The decision to execute the update process by the update agent 113 may be based on status information such as, for example, the availability of an update package. The status information may be available in the non-volatile memory 111. If it is determined that the update agent 113 needs to execute the update process, the mobile handset 107 may invoke the update agent 113.

[23] In one embodiment, the generator with nodes preprocessor 145 may generate a list of nodes when it generates an update package. Each node in the list of nodes may comprise two integers, for example, one integer indicating the address of the node, and another integer indicating the movement or the offset of the node. The node list may be incorporated within an update package and loaded into the update package repository 133. When the mobile handset 107 requests an update package from the management server 109, the update package may be delivered to the mobile handset 107 from the update package repository 133 via the management server 109. The update agent 113 in the mobile handset 107 may execute a "preprocessing" step using the nodes list, to prepare the existing or old image of the firmware/software for update. After the preprocessing, the update agent 113 may employ other information from the update package to update the firmware/software.

[24] The nodes technique for preprocessing binary images of firmware/software, may utilize "logical" alignment of elements within different versions of code such as, for example, an old version and a new version of firmware. The output of the nodes preprocessor component of the generator with nodes preprocessor 145 may include a node

list and a filter list. The node list and the filter list may be incorporated into the update package to be sent as a single package to the mobile handset 107.

[25] Fig. 1B illustrates a block diagram of exemplary changes to a binary memory image between two versions of firmware/software of a mobile handset 107, in accordance with an embodiment of the present invention. In this example, components in an old binary memory image 163 are modified and their size has changed. The old binary image 163 and the corresponding new binary image 165 have corresponding components, Obj A, 155 and 161, Obj. B, 153 and 159, and Obj. C, 151 and 157.

[26] In the example of Fig. 1B, an old binary image 163 is modified to a new binary image 165. The difference between the old binary image 163 and the new binary image 165 is a change in the size of a component, Obj. B, 153 and 159. Such change in size of one component results in a change of the starting address of another component, Obj. C, 151 and 157.

[27] In one embodiment, the node technique may include determining changes of locations of various symbols in the old binary image 163, determining "node" symbols, and predicting the location of some symbols based on relocation information gathered from the "node" symbols. In one embodiment, the nodes may be determined by comparing each symbol from the old and new binary images to determine locations of code segments in the new binary image. In a related embodiment, symbols may be identified, given that locations of such symbols may be predicted as offsets from "node" symbols. Wherein, "node" symbols appear as relocated symbols for which location may not be easily predicted, such as the symbol for the beginning of a function that may have been relocated. Determining "node" symbols may be achieved by scanning through the symbols of the old binary image 163, locating the corresponding symbols in the new binary image 165, comparing the differences in the addresses of the symbols, predicting the differences in addresses for subsequent symbols, identifying symbols with offsets that were not predicted, and using those identified symbols as "node" symbols. Determining "node" symbols may also include determining the range of addresses where offset prediction or address differences work, and capturing and storing such information as filter information. Filter information, for example, may consist of entries

wherein each entry may be a 2-tuple comprising a location and a range value as shown hereinafter in an example.

[28] The update agent 113 may employ the nodes information and filter information provided in an update package during a preprocessing phase to prepare the old binary image 163 of firmware/software for updates. The update agent 113 may then employ instructions and data provided in the update package to update the old binary image 163 to a new binary image 165.

[29] In one embodiment, the generator with nodes preprocessor 145 may employ a nodes preprocessing phase, a residue technique, and a genetic algorithm (GA) based, bank-order technique. The residue technique may be utilized to reduce the size of a generated update package. The genetic algorithm (GA) based, bank-order technique may be utilized to determine an optimum bank order.

[30] Fig. 2 illustrates a flow diagram of the operation of an exemplary generator with nodes preprocessor generating an update package using two versions of a firmware/software, in accordance with an embodiment of the present invention. The method may be used in the operation of a generator such as the generator with nodes preprocessor 145 of Fig. 1A. At the start (205), the nodes preprocessor of the generator with nodes preprocessor converts symbol information into distance information. The distance information may be collected in a file and processed during the NodeGen phase in (207).

[31] NodeGen (207), is a technique for determining nodes, as described hereinabove, and a list of nodes based on comparing address locations of symbols in the old and new binary images of firmware/software.

[32] Following the NodeGen (207), the pre-predict phase begins with instruction prediction (209), followed by a block hunting activity (211). The instruction prediction (209) may employ a disassembler to help identify instructions and fix address locations of pointers and other types of addresses. The instruction prediction (209) may also generate a subset of filter information that may be used to modify the old image of firmware/software to look like the new image. The block hunting activity (211), may then attempt to fix data by processing

sections of binary code that could not be determined to be instructions and may appear to be data. At the end of the block hunting activity (211), filter information (213), may be generated. The instruction prediction (209), and the block hunting activity (211), both produce filter information that may be collected and processed at block 213. For example, a filter entry may convey information regarding a node location and an address range over which prediction of addresses and/or data was successful, as shown hereinafter in an example. As a result, the end of the address range specified by a filter entry may be a location where an incorrect prediction may have been encountered.

[33] The preprocessor may perform a predict phase (215), where instruction prediction (217) may be performed followed by block hunting (219). The instruction prediction (217) may employ a subset of the filter information (213) that may have been generated by instruction prediction (209) in the pre-predict phase. Similarly, the block hunting (219) may employ a subset of the filter information (213) that may have been generated by block hunting (211) in the pre-predict phase.

[34] Following the predict phase (215), the generator may execute a raw generation process (221) to generate an update package. The generated update package is then outputted at the end block 223. The update package may be supplemented with data that may comprise a MemSection data, ValidateSection data, nodes information and filter information.

[35] In one embodiment, the predict phase (215) may be bypassed and the pre-predict phase followed, instead, by the processing of the filter information (213). The generation of update packages (221) may then be performed, and the update packages outputted at block 223.

[36] Fig. 3 illustrates a flow diagram of the operation of another exemplary generator with nodes preprocessor generating an update package using two versions of a firmware/software, in accordance with an embodiment of the present invention. At the start (305), the nodes preprocessor of the generator with nodes preprocessor may convert symbol information to

distance information. The distance information may be collected into a file and processed during the next phase (307), or the NodeGen phase.

[37] NodeGen (307), is a technique for determining nodes and a list of nodes based on comparing address locations of symbols in the old and new binary images of firmware/software, as described hereinabove.

[38] Following the NodeGen (307), the preprocessor may conduct instruction pre-prediction (309) as part of an instruction pre-predict activity. The preprocessor may also generate and collect filter information for filter 1 (313). The instruction pre-prediction (309) may employ a disassembler to assist in identifying instructions and fixing different types of addresses, such as address locations of pointers. Following instruction pre-prediction (309), the preprocessor may conduct instruction prediction (311), where the preprocessor may apply the generated and collected filter information to the old binary image of the firmware/ software, resulting in a "partially" modified old binary image.

[39] The instruction prediction (311) may then be followed by a pre-block hunting activity (315). During the pre-block hunting activity(315), additional filter information may be generated and collected for filter 2 (323). The block hunting activity (317) may utilize the generated and collected filter information on the "partially" modified old binary image of the firmware/software, to create a modified old binary image.

[40] Once the modified old binary image of the firmware/software is created, the generator may execute a raw generation process (319) to generate an update package. The update package is generated based on the modified old binary image and the new binary image of the firmware/software. The generated update package is then outputted (321). The update package may be supplemented with data that may comprise a MemSection data, ValidateSection data, nodes information and filters 1 (313) and filter 2 (323) information.

[41] The update package generation by the generator with nodes preprocessor may be used for mobile handsets and other mobile electronic devices, such as devices that contain code or data in non-volatile memory which may require updates to newer or different versions.

[42] Below is an example of distance information as discussed hereinabove, in a list format. Each row of the list contains four entries with a space between entries as a delimiter. The first entry indicates an address or location of a symbol, the second entry is the distance to the next symbol, the third entry is the code type, and the last entry is the name of the symbol associated, such as the name of a function, the name of a global variable, etc. Thus, the address 0x502C0000, for example, has a label BFLeadingSymbol, the symbol that follows it is a distance of 0x00046054 after it.

0x502C0000 0x00046054 C BFLeadingSymbol

0x50306054 0x000000A8 C \Epoc32\Release\WD2\urel\ekern.exe _E32Startup

0x503060FC 0x00000004 C \Epoc32\Release\WD2\urel\ekern.exe global constructors keyed to ImpAsic::Psu

0x50306100 0x00000064 C \Epoc32\Release\WD2\urel\ekern.exe ImpDma::Init1 (void)

0x50306164 0x00000010 C \Epoc32\Release\WD2\urel\ekern.exe ImpDma::Init3 (void)

0x50306174 0x00000008 C \Epoc32\Release\WD2\urel\ekern.exe ImpDma::MaxBlockSize (void)

0x5030617C 0x0000011C C \Epoc32\Release\WD2\urel\ekern.exe ImpDma::GetChannel (ImpDma::TdmaDevice)

[43] Also, below is an exemplary list of Node Information as discussed hereinabove. Each row in the list has two entries. The first entry indicates an address location of a symbol considered to be a node, and the second entry is a distance information or offset from the previous node. Hence, the first node of this example resides at location 0x502C000, which is the same in both the old and new images. Whereas the next node, located at 0x509081B4 is offset by 48 from the corresponding node in the old image.

0x502C0000 0

0x509081B4 48

0x50908434 56

0x50908660 64

0x50908908 88

0x50908B88 260

0x50908EE4 276

0x5090925C 292

0x50909994 304

[44] Also, below is an exemplary list of Filter Information as discussed hereinabove. Each row in the list has two entries. The first entry indicates an address location of a symbol considered to be a node, and the second entry is a distance or range information from that location over which prediction was successful. In this example the filter information indicates that a node symbol is located at the address 0x502FC0F0, and the following 0x00000058 symbols were predicted correctly by the process, however, the symbol at the location 0x00000058 later, i.e. the symbol at location 0x502FC148, did not match. The next correctly predicted symbol was at location 0x502FC174, and all the symbols in a range of 0x00003114 following that were predicted correctly, and so on.

0x502FC0F0 0x00000058

0x502FC174 0x00003114

0x502FF2AC 0x0000085C

0x502FFB28 0x00003E08

0x5030395C 0x0000165C

0x50304FE0 0x00000074

0x50305078 0x0004EC74

0x50362958 0x00070980

0x503D369C 0x0010E8F8

0x505017CC 0x000021AC

0x50503D9C 0x00058224

0x5057C634 0x000924E4

0x5060F468 0x00041E8C

0x5066145C 0x00008344

0x5066B74C 0x000249C8

0x506BA754 0x000C69D0

0x507DB028 0x00007004

[45] While the present invention has been described with reference to certain embodiments, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted without departing from the scope of the present invention. In addition, many modifications may be made to adapt a particular situation or material to the teachings of the present invention without departing from its scope. Therefore, it is intended that the present invention not be limited to the particular embodiment disclosed, but that the present invention will include all embodiments falling within the scope of the appended claims.